Building Simulation 2003

Eighth International IBPSA Conference
Eindhoven, Netherlands
August 11-14, 2003

COMPARISON OF A GENERALIZED PATTERN SEARCH AND A GENETIC ALGORITHM OPTIMIZATION METHOD

Michael Wetter[1] and Jonathan Wright[2].
[1]Simulation Research Group, Building Technologies Department,
Environmental Energy Technologies Division, Lawrence Berkeley National Laboratory,
Berkeley, CA 94720, USA.
[2]Department of Civil and Building Engineering, Loughborough University,
Loughborough, Leicestershire, LE11 3TU, UK.

ABSTRACT

Building and HVAC system design can significantly improve if numerical optimization is used. However, if a cost function that is smooth in the design parameter is evaluated by a building energy simulation program, it usually becomes replaced with a numerical approximation that is discontinuous in the design parameter. Moreover, many building simulation programs do not allow obtaining an error bound for the numerical approximations to the cost function. Thus, if a cost function is evaluated by such a program, optimization algorithms that depend on smoothness of the cost function can fail far from a minimum.

For such problems it is unclear how the Hooke-Jeeves Generalized Pattern Search optimization algorithm and the simple Genetic Algorithm perform. The Hooke-Jeeves algorithm depends on smoothness of the cost function, whereas the simple Genetic Algorithm may not even converge if the cost function is smooth. Therefore, we are interested in how these algorithms perform if used in conjunction with a cost function evaluated by a building energy simulation program.

In this paper we show what can be expected from the two algorithms and compare their performance in minimizing the annual primary energy consumption of an office building in three locations. The problem has 13 design parameters and the cost function has large discontinuities.

The optimization algorithms reduce the energy consumption by 7% to 32%, depending on the building location. Given the short labor time to set up the optimization problems, such reductions can yield considerable economic gains.

INTRODUCTION

We compare the Hooke-Jeeves optimization algorithm (HJ algorithm), which is a member of the family of Generalized Pattern Search algorithms (GPS algorithms), and the simple Genetic Algorithm (sGA) on a theoretical level and by using numerical experiments. We will consider problems where the cost function, such as annual energy use, is computationally expensive, defined on continuous design parameters, and evaluated by the whole-building energy simulation program EnergyPlus (Crawley et al. 2001). A similar discussion applies if other system simulation programs, such as TRNSYS or COMIS, are used to evaluate the cost function. Such programs replace the cost function with a numerical approximation that is discontinuous in the design parameters. Furthermore, their solvers are usually implemented in such a way that establishing error bounds is not possible. In these situations optimization can only be applied heuristically, and it is not clear how the HJ algorithm and the sGA perform if used in conjunction with such discontinuous cost functions.

First, we define the optimization problem and discuss the properties of the cost function. Next, we describe what can be expected from the optimization algorithms and show why the two algorithms are likely to yield different results. Then, we present numerical experiments that compare the performance of the algorithms. We analyze the observed numerical problems and close by proposing the use of a hybrid optimization algorithm.

PROBLEM STATEMENT

We consider problems of the form

$$\min_{x \in \mathbf{X}} f(x), \tag{1a}$$

where $x \in \mathbf{X}$ is the vector of design parameters, $f\colon \mathbf{X} \to \mathbb{R}$ is the cost function, and $\mathbf{X} \subset \mathbb{R}^n$ is the constraint set, defined as

$$\mathbf{X} \triangleq \left\{ x \in \mathbb{R}^n \mid l^i \le x^i \le u^i, \quad i \in \{1,\ldots,n\} \right\}, \tag{1b}$$

with $-\infty \le l^i < u^i \le \infty$, for all $i \in \{1,\ldots,n\}$.

In building design and heating, ventilation, and air-conditioning (HVAC) system optimization, evaluating the cost function usually requires solving a differential algebraic system of equations. Under appropriate assumptions, such systems have a unique solution that is continuously differentiable in the design parameter (Wetter and Polak 2003).

However, this solution must usually be approximated numerically. In approximating the solution, EnergyPlus uses adaptive variations in solver iterations and adaptive integration meshes. For such solution methods, a perturbation of the design parameter can cause a change in the sequence of solver iterations, which causes the cost function to be discontinuous. Thus, when a smooth cost function, defined on the design parameters, is evaluated by EnergyPlus, it becomes replaced with an approximation that fails to be even continuous.

Discontinuities can cause erratic behavior of optimization algorithms. For ease of explanation, consider the cost function to be $f(x) = F(x) + e(x)$, where $f(x)$ is the numerical approximation of the cost obtained by simulation, $F(x)$ is the value of the exact, smooth cost function, which is not available, and $e(x)$ is the unknown approximation error. Suppose that there exists a scalar $\delta > 0$ such that $|e(x)| < \delta$, for all $x \in \mathbf{X}$. Clearly, we are interested in finding the minimum of $F(\cdot)$ and not the minimum of $f(\cdot)$ because $e(\cdot)$ is of no physical significance. However, the optimization algorithm can only use values of $f(\cdot)$, and a difference in $f(\cdot)$ between two points $x', x'' \in \mathbf{X}$ need not be a difference in $F(\cdot)$ if $|f(x') - f(x'')| < 2\delta$. Thus, if $|f(x') - f(x'')| < 2\delta$, then an optimization algorithm may make a wrong decision about the descent of $F(\cdot)$. In such situations the algorithm may behave erratically, which may result in failure or in many more function evaluations.

If high precision simulations can be used, then δ can be made small, which improves the convergence of the optimization algorithms. However, in EnergyPlus many precision parameters are fixed at compile time. Adjusting the different precision parameters independently can cause such a differential algebraic system of equations to diverge. Moreover, due to the way the solvers are implemented, it seems to be impossible to establish a scheme for adjusting all precision parameters and to obtain an error bound for the approximate solution. This obviously limits the potential gains that optimization offers. Therefore, we propose that simulation programs be written such that precision can be changed without recompilation. This would allow adjusting the precision of at least some solvers that are expected to cause large discontinuities in the cost function. Moreover, if the numerical solvers are implemented such that error bounds for the approximate solutions can be obtained, then one can control the error such that any Generalized Pattern Search (GPS) algorithm converges to a stationary point of $F(\cdot)$ (see Polak and Wetter 2003; Wetter and Polak 2003).

Because $f(\cdot)$ is discontinuous, it may only have an infimum (i.e., a greatest lower bound) but no minimum even if $\mathbf{X}$ is compact. Thus, to be correct, (1a) should be replaced by $\inf_{x\in\mathbf{X}} f(x)$. For simplicity, we will not make this distinction.

We will show that the cost function used in our numerical experiments is not convex and therefore can have multiple local minima. Thus, the optimization algorithms can be attracted by a local minimum that may be far from a global minimum.

ALGORITHM DESCRIPTION

Hooke-Jeeves Algorithm

The HJ algorithm is a member of the family of GPS algorithms. Torczon (1997) and Audet and Dennis (2003) proved that for problem (1) with continuously differentiable cost function having bounded level sets, any GPS method constructs a sequence of iterates that converges to a stationary accumulation point.

Let $k \in \mathbb{N}$ denote the iteration number, and let $x_k \in \mathbf{X}$ denote the current iterate. GPS algorithms have in common that after a finite number of iterations, they search for a lower cost function value than $f(x_k)$ on the points in the set

$$\mathcal{L}_k \triangleq \{x \in \mathbf{X} \mid x = x_k \pm \Delta_k s^i e_i,\ i \in \{1, \ldots, n\}\ \}, \quad (2)$$

where $\Delta_k > 0$ is a scalar called the *mesh size factor*, and $s \in \mathbb{R}^n$ is a fixed parameter that can be used to take the different scaling of the design parameter components into account. In addition, each GPS algorithm has a rule that selects a finite number of points in $\mathbf{X}$ on a mesh defined by

$$\mathbb{M}(x_0, \Delta_k) \triangleq \{x_0 + m \Delta_k s^i e_i \mid i \in \{1, \ldots, n\}, m \in \mathbb{Z}\}, \quad (3)$$

where $x_0 \in \mathbf{X}$ is the initial iterate. It is this rule that distinguishes the different GPS algorithms such as the HJ algorithm or the Coordinate Search algorithm.

If a mesh point $x' \in \mathbb{M}(x_0, \Delta_k)$ with $f(x') < f(x_k)$ has been found, then the search continues with $x_{k+1} = x'$ and $\Delta_{k+1} = \Delta_k$. Otherwise, all points in $\mathcal{L}_k$ are tested for a decrease in $f(\cdot)$. If $f(x') \geq f(x_k)$ for all $x' \in \mathcal{L}_k$, then the search continues with $x_{k+1} = x_k$ and a reduced mesh size factor, say $\Delta_{k+1} = \Delta_k/2$. Hence, the search continues on a finer mesh. The search stops if the mesh $\mathbb{M}(x_0, \cdot)$ has been refined a user-specified number of times. Thus, if Δ_{min} denotes the smallest mesh size factor, then the lowest point x^* obtained by a GPS algorithm satisfies $f(x^*) \leq f(x')$ for all $x' \in \{x \in \mathbf{X} \mid x = x^* \pm \Delta_{min} s^i e_i,\ i \in \{1, \ldots, n\}\ \}$.

See Hooke and Jeeves (1961) for a more detailed description of the HJ algorithm. See Torczon (1997) and Audet and Dennis (2003) for a convergence analysis of GPS algorithms for smooth cost functions. See Polak and Wetter (2003) and Wetter and Polak (2003) for an algorithm that controls the precision of the approximating cost function. Their algorithm ensures convergence to stationary points and can be exploited to obtain a significant reduction in computing time.

Genetic Algorithm

GA are algorithms that operate on a finite set of points, called a *population*. The different populations are interpreted as *generations*. They are derived on the principles of natural selection and incorporate operators for (1) fitness assignment, (2) selection of points for recombination, (3) recombination of points, and (4) mutation of a point.

Our GA is an implementation of the *simple GA* described by Goldberg (1989), but we use a Gray (Press et al. 1993) rather than a pure binary encoding to represent the design parameters as a concatenated string of binary numbers. In Gray coding, if two integers are adjacent, then their binary representation differs by only one bit.

For our sGA, all lower and upper bounds of $\mathbf{X}$ need to be finite. Our sGA discretizes $\mathbf{X}$ in the mesh $\mathbb{M}(l,1)$, and all operators are such that the real value of any point is an element of $\mathbb{M}(l,1) \cap \mathbf{X}$.

Some of the GA operators take as an argument a set of points and return a set of points, whereas both sets have a prescribed number of elements. To define the domain and range of these operators, we need to introduce some notation. Given a non-empty set $\mathbf{X} \subset \mathbb{R}^n$ and a non-zero $M \in \mathbb{N}$, we define $\underline{\mathbf{X}}_M$ as the set of all sequences in $\mathbf{X}$ with M elements, i.e.,

$$\underline{\mathbf{X}}_M \triangleq \big\{ \{x_i\}_{i=1}^M \mid x_i \in \mathbf{X},\ i \in \{1,\ldots,M\} \big\}. \qquad (4)$$

We define $\mathbf{B}$ as the set that contains all elements of $\mathbb{M}(l,1) \cap \mathbf{X}$ as encoded concatenated strings of binary numbers (Goldberg 1989). We define $\underline{\mathbf{B}}_M$ as the set of all sequences in $\mathbf{B}$ with M elements, i.e.,

$$\underline{\mathbf{B}}_M \triangleq \big\{ \{\chi_i\}_{i=1}^M \mid \chi_i \in \mathbf{B},\ i \in \{1,\ldots,M\} \big\}. \qquad (5)$$

We will denote by $k \in \mathbb{N}$ the generation number, by $M \in \mathbb{N}$ the population size, by $\underline{x}_k \in \underline{\mathbf{X}}_M$ the M points in the k-th generation, and by $\underline{\chi}_k \in \underline{\mathbf{B}}_M$ the binary representation of $\underline{x}_k$. For $i \in \{1,\ldots,M\}$, we will write $\underline{\chi}_{k,i}$ to denote the i-th element of $\underline{\chi}_k$. That is, $\underline{\chi}_{k,i}$ is the binary representation of a point in $\mathbb{R}^n$.

The sGA starts by generating an initial population $\underline{\chi}_0 \in \underline{\mathbf{B}}_M$ of M randomly generated points. Then, the cost function is evaluated for each point in $\underline{\chi}_k$, and a *fitness function* $\Theta\colon \mathbb{N} \times \underline{\mathbf{B}}_M \to \mathbb{R}^M$ computes for each point in $\underline{\chi}_k$ a fitness value. The fitness of a point indicates the worth of the point in relation to all other points in the population. In our implementation, a point with a low cost function value is considered to be fitter than a point with a high cost function value.

For $N \in \mathbb{N}$, with $0 < N < M$, an *elitism function* $\alpha\colon \underline{\mathbf{B}}_M \to \underline{\mathbf{B}}_N$ selects the N fittest points of the generation. These points will be put in the next generation.

Afterwards, a *selection function* $\vartheta\colon \mathbb{N} \times \underline{\mathbf{B}}_M \to \underline{\mathbf{B}}_2$ selects a pair of points. The selection is probabilistic, whereas fitter points are more likely to be selected, i.e., we use a *proportional* selection algorithm.

A *recombination function* $\phi\colon \mathbb{N} \times \underline{\mathbf{B}}_2 \times [0,1] \to \underline{\mathbf{B}}_2$ recombines the selected points to a new pair of points. We use a *single-point crossover* recombination. In a single-point crossover recombination, one crossover position is randomly selected and the bits after this position are exchanged between the two points. The probability of recombination $p_r \in [0,1]$ is an algorithm parameter.

Each recombined point is mutated by a *mutation function* $\Psi\colon \mathbb{N} \times \mathbf{B} \times [0,1] \to \mathbf{B}$, which changes the value of some bits of the binary representation. The degree of mutation depends on the generation number (Deb 2000), and the probability of mutation is an algorithm parameter. Finally, both mutated points are placed in the next generation.

Since it is difficult to check convergence for a GA, we run the GA for a prescribed number of generations.

Figure 1 shows our simple GA. The transformations from $\underline{x}_k$ to $\underline{\chi}_k$ and from $\underline{\chi}_k$ to $\underline{x}_k$ are not shown explicitly.

Comparison of the Hooke-Jeeves Algorithm and the Genetic Algorithm

The HJ algorithm does not explore the global structure of the cost function and thus can get attracted by any minimum, local or global. The HJ algorithm constructs a sequence of iterates that converge to a stationary point if the cost function is smooth and coercive. If the cost function is discontinuous, the HJ algorithm can fail at a discontinuity.

For the HJ algorithm, the design parameters must be continuous. Audet and Dennis (2000) give an extension that allows having discrete parameters.

Simple Genetic Algorithm

```
Data:    Step size $s \in \mathbb{R}^n$ for the discretization of $\mathbf{X}$;
         Probability for recombination $p_r \in [0,1]$;
         Probability for mutation $p_m \in [0,1]$;
         Number of generations $K \in \mathbb{N}$;
         Population size $M \in \mathbb{N}$;
         Number of elite points $N \in \mathbb{N}$, $0 < N < M$.
Maps:    Fitness function $\Theta\colon \mathbb{N} \times \underline{\mathbf{B}}_M \to \mathbb{R}^M$;
         Elitism function $\alpha\colon \underline{\mathbf{B}}_M \to \underline{\mathbf{B}}_N$;
         Selection function $\vartheta\colon \mathbb{N} \times \underline{\mathbf{B}}_M \to \underline{\mathbf{B}}_2$;
         Recombination function
         $\phi\colon \mathbb{N} \times \underline{\mathbf{B}}_2 \times [0,1] \to \underline{\mathbf{B}}_2$;
         Mutation function $\Psi\colon \mathbb{N} \times \mathbf{B} \times [0,1] \to \mathbf{B}$.
Step 0:  Initialize $k = 0$, $\underline{\chi}_k \in \underline{\mathbf{B}}_M$;
         For $j \in \{1, \ldots, N\}$, evaluate $f(\underline{x}_{k,j})$.
Step 1:      Compute Fitness
             For $j \in \{N+1, \ldots, M\}$, evaluate $f(\underline{x}_{k,j})$;
             Compute $\Theta(k, \underline{\chi}_k)$.
Step 2:      Select elite points
             For $j \in \{1, \ldots, N\}$, set $\underline{\chi}_{k+1,j} = \big(\alpha(\underline{\chi}_k)\big)_j$;
             Set $i = N + 1$.
Step 3:          Select pair
                 $\{\chi_a, \chi_b\} = \vartheta(k, \underline{\chi}_k)$.
Step 4:          Recombine pair
                 $\{\chi'_a, \chi'_b\} = \phi(k, \{\chi_a, \chi_b\}, p_r)$.
Step 5:          Mutate points
                 Set $\underline{\chi}_{k+1,i} = \Psi(k, \chi'_a, p_m)$;
                 If $i + 1 \le M$
                     set $\underline{\chi}_{k+1,i+1} = \Psi(k, \chi'_b, p_m)$.
Step 6:      Replace $i$ by $i + 2$;
             If $i + 1 < M$ go to Step 3.
Step 7:  Replace $k$ by $k + 1$;
         If $k < K$ go to Step 1, else stop.
```

Figure 1: Simple Genetic Algorithm used in the numerical experiments.

The sGA starts with a population of points that are randomly distributed in **X**. This reduces the risk of being trapped in a local minimum that is not global. However, little can be said about the convergence of the sGA, even on smooth cost functions. In fact, in the course of the optimization, the population of points can collapse to a small subset of **X**. In such situations, the sGA can fail far from a minimum, and such a failure is difficult to detect. Therefore, defining a stopping criterion is difficult.

Since our GA works on a fixed discrete number of points of **X**, it can also be used for problems with discrete design parameters.

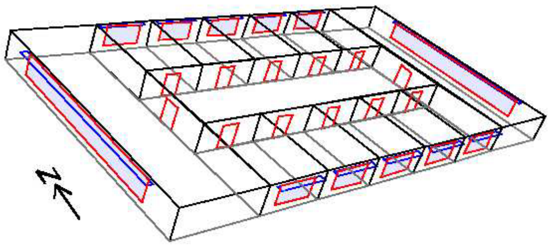


Figure 2: Simulated office floor.

NUMERICAL EXPERIMENTS

Cost Function

We attempt to solve (1) with $n = 13$ design parameters. For all examples, $f(\cdot)$ is the annual primary energy consumption for lighting, fan, cooling and heating of the mid-storey office floor shown in Figure 2. Lighting and fan electricity are multiplied by a factor of 3 and then added to the cooling and heating energy of the cooling and heating coil. $f(\cdot)$ is evaluated using an annual EnergyPlus 1.0.3 simulation. In the building shown in Figure 2, the west and east facing zone, the interior zone, and one north and one south facing zone are being simulated. The energy consumption and the system load of the north and south facing zones are multiplied by a factor of 5 to account for the non-simulated zones which are assumed to have the same conditions. All zones have daylighting control. The simulated HVAC system is a VAV system with DX coil and outside-air economizer. The heating and cooling coil capacities and the air flow rates are auto-sized by EnergyPlus. The exterior walls have a U-value of $0.25\,\mathrm{W/(m^2K)}$ and consist of (listed from outside to inside) 1 cm wood siding, 10 cm insulation and 20 cm concrete. The ceiling and floor consist of carpet, 5 cm concrete, insulation and 18 cm concrete. Interior walls are 12 cm brick. Both windows are low-emissivity double pane with Krypton gas fill and exterior shading device. We use TMY2 weather data for Houston Intercontinental (TX), Chicago O'Hare (IL), and Seattle Tacoma (WA).

Table 1 shows the design parameters, their lower and upper bound, and their step size used in the optimizations. w_i, $i \in \{N, W, E, S\}$, is a parameter that scales linearly the window width and height. The subscripts indicate north, west, east, and south, respectively. (The location and shape of the windows are used in the daylighting calculations.) For the north and south windows, a value of 0 corresponds to a window that covers 13.6% of the facade area and 1 corresponds to 64.8%. For the west and east windows, a value of 0 corresponds to a window

Table 1: *Variable symbols, base design (used as initial value for the HJ algorithm), lower bound, upper bound and step size of the design parameter, and best obtained iterate of the HJ algorithm and the sGA. The variable symbols are explained in the text. The last rows show the corresponding cost function values and the obtained reduction in cost.*

variable symbols	x_b	l	u	s	best iterate x^*, Houston HJ	best iterate x^*, Houston sGA	best iterate x^*, Chicago HJ	best iterate x^*, Chicago sGA	best iterate x^*, Seattle HJ	best iterate x^*, Seattle sGA
w_N	0.5	0	1	0.05	0.03	0.6	0.00	0.00	0.51	0.60
w_W	0.5	0	1	0.05	0.12	0.05	0.16	0.20	0.38	0.45
o_W	0.5	0	1	0.05	0.99	1.00	0.99	0.95	0.99	1.00
w_E	0.5	0	1	0.05	0.04	0.10	0.23	0.30	0.58	0.80
o_E	0.5	0	1	0.05	1.00	1.00	0.99	1.00	0.96	1.00
w_S	0.5	0	1	0.05	0.98	0.7	0.99	1.00	0.99	1.00
o_S	0.5	0	1	0.05	0.00	1.00	0.00	0.55	0.89	0.90
s_W	200	100	600	25	600	600	484	400	381	300
s_E	200	100	600	25	575	600	348	250	288	225
s_S	200	100	600	25	600	600	594	575	369	450
T_u	22	20	25	0.25	22.75	24.50	21.00	24.25	25.00	25.00
T_i	22	20	25	0.25	22.31	22.75	24.97	24.75	25.00	25.00
T_d	15	12	18	0.25	12.94	12.25	12.66	14.50	12.00	12.25
$f(x_b)$ in kWh/(m^2 a)					187.6		139.6		116.4	
$f(x^*)$ in kWh/(m^2 a)					144.3	128.4	129.6	119.6	86.05	86.23
Obtained reduction in %					23.1	31.6	7.2	14.4	26.1	25.9

that covers 20.4% of the facade area and 1 corresponds to 71.3%. o_i, $i \in \{W, E, S\}$, is a parameter that scales the depth of the window overhangs. A value of 0 means that the window overhang depth is 0.05 m (measured from the facade), and 1 means that the window overhang depth is 1.05 m. s_i, $i \in \{W, E, S\}$, is the set point for the shading device in W/m^2. If the direct plus diffuse solar radiation incident on the window exceeds s_i, then an external shading device with a transmittance of 0.5 is activated. T_i, $i \in \{u, i\}$, is the set point for the zone air temperature for night cooling in summer and winter, respectively, in °C. T_d is the cooling design supply air temperature that is used for the HVAC system sizing in °C. The column with header x_b shows the values of the design parameters for the base design, l and u are the lower and upper bounds, and s is the step size of the design parameter.

The base design is for all locations the same. No simulations have been done to select the values of x_b.

Algorithm Parameters

For the HJ algorithm we use the initial iterate $x_0 = x_b$, the step size s^i, $i \in \{1, \ldots, n\}$, as shown in Table 1, the initial mesh size factor $\Delta_0 = 1$ and 4 step reductions in which we set $\Delta_{k+1} = \Delta_k/2$. Hence, $\Delta_{min} = 1/16$, and the best point x^* found by the HJ algorithm satisfies $f(x^*) \leq f(x')$, for all $x' \in \{x \in \mathbf{X} \mid x = x^* \pm (s^i/16)\, e_i,\ i \in \{1, \ldots, n\}\ \}$.

For the sGA we select a population size $M = 15$, one elite point $N = 1$, a probability for recombination $p_r = 1$, a probability for mutation $p_m = 0.02$, and a number of generations $K = 50$. Thus, the maximum number of function evaluations is $K(M - N) + N = 701$. The choice of population size M, probability of mutation p_m and recombination p_r is dependent on the characteristics of the optimization problem, as well as the type of genetic algorithm operators (Deb 2001)[1]. Commonly used population sizes M range from 5 to 100, the recombination probability p_r from 0.7 to 1.0, and the mutation probability p_m from 0.001 to 0.05. In general, a large number of design parameters n, a large number of local minima and a pronounced non-linearity of the cost function require a larger population size M and a higher probability for recombination p_r and mutation p_m. Increasing the population size M increases the diversity of the solutions and the likelihood of finding the global minimum. The diversity in the population can also be maintained through a high probability of mutation p_m. For our numerical experiments, we selected a small population size M because the number of design parameters is small and because we expected the cost function to have no significant local minima. The choice of a small population size

[1]Although the primary focus of this book is on multi-objective evolutionary optimization, it includes a general and thorough discussion of the issues surrounding the choice of population size, and the probability of mutation and recombination.

M was balanced by a high probability of recombination p_r and mutation p_m. Small population sizes have also been used successfully in the solution of other small scale building design optimization problems ($M = 5$) (Caldas and Norford 2002), although larger scale highly constrained optimization problems have required larger population sizes and a higher number of generations ($M = 80$, $K = 1000$) (Wright and Farmani 2001). Further research is required to determine the most effective population size and probability of recombination and mutation for building design optimization problems, particularly in view of the need to limit the number of cost function evaluations which are typically computationally intensive.

Comparison of the Optimization Results

We used the optimization program GenOpt 2.0α (Wetter 2001)[2] which called EnergyPlus 1.0.3 iteratively to evaluate the cost function. Both programs were run on a 2.4 GHz computer under Windows XP. All optimizations took about 24 hours computation time.

Table 1 shows that significant energy saving were obtained for all locations. For Houston and Chicago, the sGA found a solution with lower cost than the HJ algorithm. For these optimizations, the HJ algorithm may have failed at a discontinuity of the cost function, or may have been attracted by a local minimum with higher cost function value than the best iterate of the sGA.

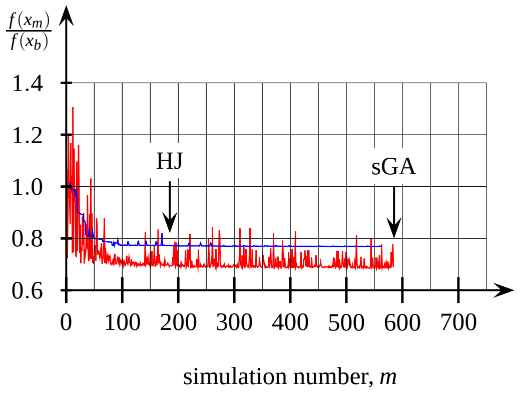
In Figure 3, $m \in \mathbb{N}$ denotes the simulation number and $f(x_m)/f(x_b)$ is the ratio of the m-th cost function value and the cost function value of the base design. The sGA requires less than the maximum number of simulations since some points were generated more than once. In such situations, no simulation is done. Within 100 simulations, the sGA and the HJ algorithm come close to its minimum cost function value, except for the Seattle case, in which the HJ algorithm requires about 200 simulations to come close to its minimum cost function value.

In Figure 4, we show which part of the cost function is due to lighting, heating, fan, and cooling energy for the base design and for the best iterate of the sGA for Houston.

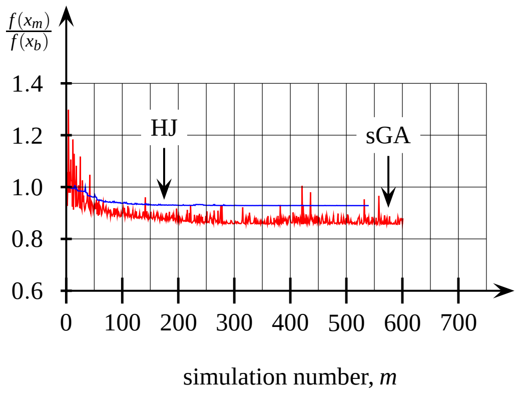
Discontinuities in the Cost Function

Now we will show that the cost function has large discontinuities. Let $x_H^* \in \mathbb{R}^n$ denote the iterate with the lowest cost function value obtained by the HJ algorithm for Houston, and let x_G^* denote the iterate with lowest cost function value obtained by the sGA for Houston. In Figure 5, we show, for $\lambda \in$

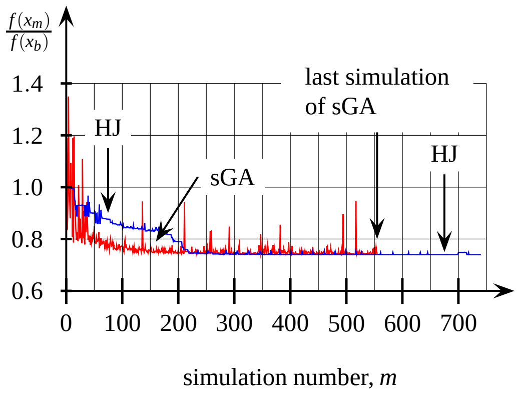
[2]Work for implementing a Genetic Algorithm for GenOpt 2.0 is currently in progress.



(a) Houston, TX



(b) Chicago, IL



(c) Seattle, WA

Figure 3: *Normalized cost function value as a function of the simulation number for the different algorithms and building locations.*

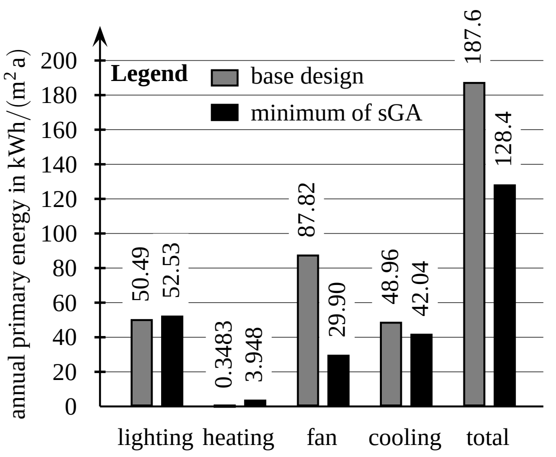


Figure 4: *Primary energy consumption for lighting, heating, fan, cooling, and total primary energy consumption for Houston, TX.*

$\{0, 0.01, \ldots, 1\}$, the ratio $f(x(\lambda))/f(x(0))$, where

$$x(\lambda) \triangleq x_H^* + \lambda\,(x_G^* - x_H^*)\,. \qquad (6)$$

That is, $x(0) = x_H^*$ and $x(1) = x_G^*$. The graph shows discontinuities on the order of 3% of the cost function value. Such large discontinuities can cause the HJ algorithm to fail far from a minimum.

We believe that the discontinuities are caused in computations of time-independent values, such as values that are used to auto-size the HVAC system. To support this claim, we show in Figure 5 the volume flow rate as determined by auto-sizing the heating system for all five north zones $\dot{V}_N(x)$. Some perturbations of x change $\dot{V}_N(x)$ by as much as 25%. This may be because EnergyPlus may fail to solve the zone heat balance equations after a set point change in the zone air temperature, or because the time integration mesh for the heating design day may change from one simulation to the next.

Proposal for a Hybrid Optimization Algorithm

Since the sGA searches on the mesh $\mathbb{M}(l, 1)$ and stops after a finite number of iterations, it would be easy to combine the sGA with the Coordinate Search algorithm to produce a GPS algorithm that uses the sGA for the global search and uses the Coordinate Search for the local search. Then, the global exploration of the sGA reduces the risk of getting attracted by a local minimum which is not global, and the Coordinate Search enables clear convergence statements in domains where the cost function is smooth.

CONCLUSION

By using optimization, we achieved energy savings between 7% and 32%, depending on the building location. Since the labor to connect

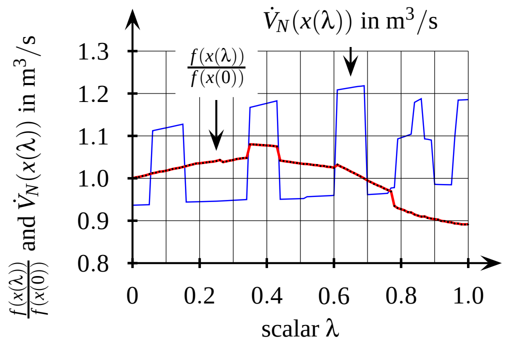


Figure 5: *Cost function value $f(x(\lambda))$ normalized against $f(x(0))$ and volume flow rate as computed by the heating system auto-sizing for all five north zones $\dot{V}_N(x(\lambda))$. The functions are evaluated on the line between the minimum point obtained by the HJ algorithm (at $\lambda = 0$) and the minimum point obtained by the sGA (at $\lambda = 1$) for Houston, TX.*

GenOpt to EnergyPlus and to define the optimization problems was only a few hours, this shows that optimization can yield considerable economic gains. Clearly, such a design that depends nonlinearly on 13 design parameters could not have been achieved without using optimization algorithms.

For all test cases, our sGA uses a number of function evaluations that is comparable to the HJ algorithm. However, the speed of convergence of the HJ algorithm may have been reduced due to large discontinuities in the cost function.

In two of the three problems, the sGA yields a design with lower cost function value than the HJ algorithm. This can be because the cost function has discontinuities or because it may have several local minima.

Both algorithms are good candidates for solving the examined problems, but the computation time could be reduced and the accuracy of the solution improved if the simulation program allowed controlling the error of the approximations to the cost function.

ACKNOWLEDGMENTS

This research was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of the Building Technologies Program of the U.S. Department of Energy, under Contract No. DE-AC03-76SF00098, and through a Foresight Award from the UK Royal Academy of Engineering.

REFERENCES

Audet, Charles, and J. E. Dennis, Jr. 2000. "Pattern Search Algorithms for Mixed Variable Programming." *SIAM Journal on Optimization* 11 (3): 573–594.

———. 2003. "Analysis of Generalized Pattern Searches." *SIAM Journal on Optimization* 13 (3): 889–903.

Caldas, Luisa Gama, and Leslie K. Norford. 2002. "A design optimization tool based on a genetic algorithm." *Automation in Construction* 11 (2): 173–184.

Crawley, Drury B., Linda K. Lawrie, Frederick C. Winkelmann, W.F. Buhl, Y. Joe Huang, Curtis O. Pedersen, Richard K. Strand, Richard J. Liesen, Daniel E. Fisher, Michael J. Witte, and Jason Glazer. 2001. "EnergyPlus: creating a new-generation building energy simulation program." *Energy and Buildings* 33 (4): 443–457.

Deb, Kalyanmoy. 2000. "An efficient constraint handling method for genetic algorithms." *Computer methods in applied mechanics and engineering* 186:311–338.

———. 2001. *Multi-Objective Optimization using Evolutionary Algorithms.* Interscience Series in Systems and Optimization. John Wiley & Sons, Inc. ISBN: 1-58603-256-9.

Goldberg, D.E. 1989. *Genetic Algorithm in Search, Optimization, and Machine Learning.* Adisson-Wesley.

Hooke, R., and T. A. Jeeves. 1961. "'Direct search' solution of numerical and statistical problems." *J. Assoc. Comp. Mach.* 8 (2): 212–229.

Polak, Elijah, and Michael Wetter. 2003. "Generalized Pattern Search Algorithms with Adaptive Precision Function Evaluations." Technical Report LBNL-52629, Lawrence Berkeley National Laboratory, Berkeley, CA.

Press, W. H., B. P. Flannery, S. A. Tuekolsky, and W. T. Vetterling. 1993. Chapter 20 of *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press.

Torczon, Virginia. 1997. "On the Convergence of Pattern Search Algorithms." *SIAM Journal on Optimization* 7 (1): 1–25.

Wetter, Michael. 2001, August. "GenOpt – A Generic Optimization Program." Edited by R. Lamberts, C. O. R. Negrão, and J. Hensen, *Proc. of the 7-th IBPSA Conference*, Volume I. Rio de Janeiro, Brazil, 601–608.

Wetter, Michael, and Elijah Polak. 2003, August. "A convergent optimization method using pattern search algorithms with adaptive precision simulation." *To appear: Proc. of the 8-th IBPSA Conference*. Eindhoven, NL.

Wright, Jonathan, and Raziyeh Farmani. 2001, August. "The Simultaneous Optimization of Building Fabric Construction, HVAC System Size, and the Plant Control Strategy." Edited by R. Lamberts, C. O. R. Negrão, and J. Hensen, *Proc. of the 7-th IBPSA Conference*, Volume I. Rio de Janeiro, Brazil, 865–872.

NOMENCLATURE

Conventions

1. Vectors are always column vectors, and their elements are denoted by superscripts.
2. Elements of a set or a sequence are denoted by subscripts.
3. $f(\cdot)$ denotes a function where $(\cdot)$ stands for the undesignated variables. $f(x)$ denotes the value of $f(\cdot)$ for the argument x. $f\colon A \to B$ indicates that the domain of $f(\cdot)$ is in the space A, and that the image of $f(\cdot)$ is in the space B.
4. We say that a function $f : \mathbb{R}^n \to \mathbb{R}$ is once continuously differentiable on a set $\mathbf{S} \subset \mathbb{R}^n$ with respect to $x \in \mathbf{S}$ if $f(\cdot)$ is defined on $\mathbf{S}$, and if $f(\cdot)$ has a continuous derivative on $\mathbf{S}$.
5. For $x^* \in \mathbb{R}^n$ and $f\colon \mathbb{R}^n \to \mathbb{R}$ continuously differentiable, we say that x^* is stationary if $\nabla f(x^*) = 0$.
6. We say that a function $f : \mathbb{R}^n \to \mathbb{R}$ is coercive if $\lim_{k\to\infty} f(x_k) = \infty$ for every sequence $\{x_k\}_{k=0}^{\infty} \subset \mathbb{R}^n$ such that $\|x_k\| \to \infty$, as $k \to \infty$, for some norm $\|\cdot\|$.
7. We denote by $\{e_i\}_{i=1}^n$ the unit vectors in $\mathbb{R}^n$.

Symbols

$f(\cdot)$	cost function
l	lower bound of the design parameter
n	dimension of the design parameter
u	upper bound of the design parameter
x	design parameter
$a \in A$	a is an element of A
$A \subset B$	A is a subset of B
$A \cap B$	intersection of the sets A and B
$\mathbf{X}$	feasible set of the design parameter
$\mathbb{R}$	set of real numbers
$\mathbb{Z}$	set of integers
$\mathbb{N}$	set of natural numbers
Δ_k	mesh size factor at the k-th iteration
$\triangleq$	equal by definition